Preprints der Zeitschrift *BIBLIOTHEK – Forschung und Praxis*, 2020, AR 3352 **OCR-D**

AR 3352

OCR-D – Koordinierte Förderinitiative zur Weiterentwicklung von Verfahren der Optical Character Recognition (OCR)

OCR-D kompakt: Ergebnisse und Stand der Forschung in der Förderinitiative

Zusammenfassung: Bereits seit einigen Jahren werden große Anstrengungen unternommen, um die im deutschen Sprachraum erschienenen Drucke des 16.-18. Jahrhunderts zu erfassen und zu digitalisieren. Deren Volltexttransformation konzeptionell und technisch vorzubereiten, ist das übergeordnete Ziel des DFG-Projekts OCR-D, das sich mit der Weiterentwicklung von Verfahren der Optical Character Recognition befasst. Der Beitrag beschreibt den aktuellen Entwicklungsstand der OCR-D-Software und analysiert deren erste Teststellung in ausgewählten Bibliotheken.

Schlüsselwörter: OCR, VD, Software Test

OCR-D Compact: Results and State of Research in the Funding Initiative

Abstract: For some years, great efforts have been made to record and digitise the printed works of the 16th-18th centuries published in the German-speaking countries. Preparing their full-text transformation conceptually and technically is the overall goal of the DFG-project OCR-D, which is concerned with the further development of optical character recognition methods. The article describes the current state of development of the OCR-D software and analyses its initial testing in selected libraries.

Keywords: OCR, VD, software test

1 Hintergrund[1]

Mit den Verzeichnissen der im deutschen Sprachraum erschienenen Drucke des 16.-18. Jahrhunderts (VD16, VD17, VD18) wird eine retrospektive Nationalbibliografie des frühneuzeitlichen Schriftguts aus dem deutschsprachigen Raum erstellt. Um der Forschung die Zugänglichkeit zu diesen Texten zu

[1] Der vorliegende Beitrag wurde mit Ausnahme des Kapitels 3.3 verfasst von Konstantin Baierer, Matthias Boenig, Dr. Elisabeth Engl, Volker Hartmann, Clemens Neudecker, Reinhard Altenhöner, Dr. Alexander Geyken, Dr. Johannes Mangei und Dr. Rainer Stotzka.

erleichtern, wurden und werden große, konzertierte Anstrengungen unternommen, Volldigitalisate oder Schlüsselseiten zu den einzelnen verzeichneten Titeln digital bereitzustellen.

Mit Blick auf die Entwicklungen und neuen Möglichkeiten im Bereich der Optical Character Recognition (OCR) haben Experten im März 2014 im Rahmen eines DFG-Workshops die Volltexttransformation der VD als ambitioniertes, aber erreichbares Ziel eingeschätzt. Volltexte zum Zweck der Volltextsuche und Weiterbearbeitung, bspw. mit Werkzeugen der Digital Humanities, verfügbar zu machen, ist ein großes Desiderat der Forschung, das durch eine koordinierte Förderinitiative zu bearbeiten ist.

OCR ist ein umfassender Prozess, der typischerweise eine Abfolge von mehreren Schritten im Workflow beinhaltet: Neben der reinen Erkennung von Buchstaben und Wörtern werden Techniken wie die Vorverarbeitung (Bildoptimierung und Binarisierung), die Layoutanalyse (Erkennung und Klassifizierung von Strukturmerkmalen wie Überschriften, Absätzen usw.) und die Nachbearbeitung (Fehlerkorrektur) angewendet. Während die meisten dieser Schritte auch von der Nutzung Tiefer Neuronaler Netze profitieren können, sind bisher kaum freie und offene Standardwerkzeuge und damit verbundene Best Practices entstanden. Die Volltexterkennung historischer Dokumente wird insbesondere durch deren große Variabilität in Schriftart, Layout, Sprache und Orthographie erschwert.

2 Ziele und Aufbau des OCR-D-Projekts

Hier setzt das DFG-geförderte Projekt OCR-D an, dessen Hauptziel die konzeptionelle und technische Vorbereitung der Volltexttransformation der VD ist. Die Aufgabe der automatischen Volltexterkennung wird in ihre einzelnen Prozessschritte zerlegt, die in der Open Source OCR-D-Software nachvollzogen werden können. Dies ermöglicht es, optimale Workflows für die zu prozessierenden alten Drucke zu erstellen und damit wissenschaftlich verwertbare Volltexte zu generieren.

Dazu wurde ein Koordinationsprojekt aus der Berlin-Brandenburgischen Akademie der Wissenschaften, der Herzog-August Bibliothek Wolfenbüttel, der Staatsbibliothek zu Berlin sowie dem Karlsruher Institut für Technologie gebildet.[2] Dieses identifizierte in der ersten Projektphase Entwicklungsbedarfe, die in der zweiten Projektphase von insgesamt acht OCR-D-Modulprojekten bearbeitet wurden.

[2] In der ersten Projektphase war auch die Bayerische Staatsbibliothek München Teil des Koordinierungsprojekts, das KIT ist dagegen erst seit der zweiten Projektphase an OCR-D beteiligt.

Volltexterkennung wird als ein komplexer Prozess aufgefasst, der neben der eigentlichen Texterkennung mehrere vor- und nachgelagerte Schritte miteinschließt (vgl. Abb. 1). Zunächst wird ein Bilddigitalisat im Preprocessing für die Texterkennung aufbereitet, indem es nach Bedarf in ein Schwarz-Weiß-Bild umgewandelt (Binarization), zugeschnitten (Cropping), begradigt (Deskewing), entzerrt (Dewarping) und von Flecken bereinigt (Despeckling) wird. Im Anschluss erfolgt die Layouterkennung, die die Textbereiche einer Seite bis auf Zeilenebene identifiziert. Besonders die Erkennung der Zeilen bzw. der Grundlinie ist wichtig für die anschließende Texterkennung, die in aktuellen Ansätzen auf Neuronalen Netzen beruht. Danach werden die einzelnen Strukturen bzw. Elemente des Dokuments ihrer typografischen Funktion nach klassifiziert und das OCR-Ergebnis ggf. in der Nachkorrektur verbessert, bevor es in Repositorien zur Langzeitarchivierung überführt wird.

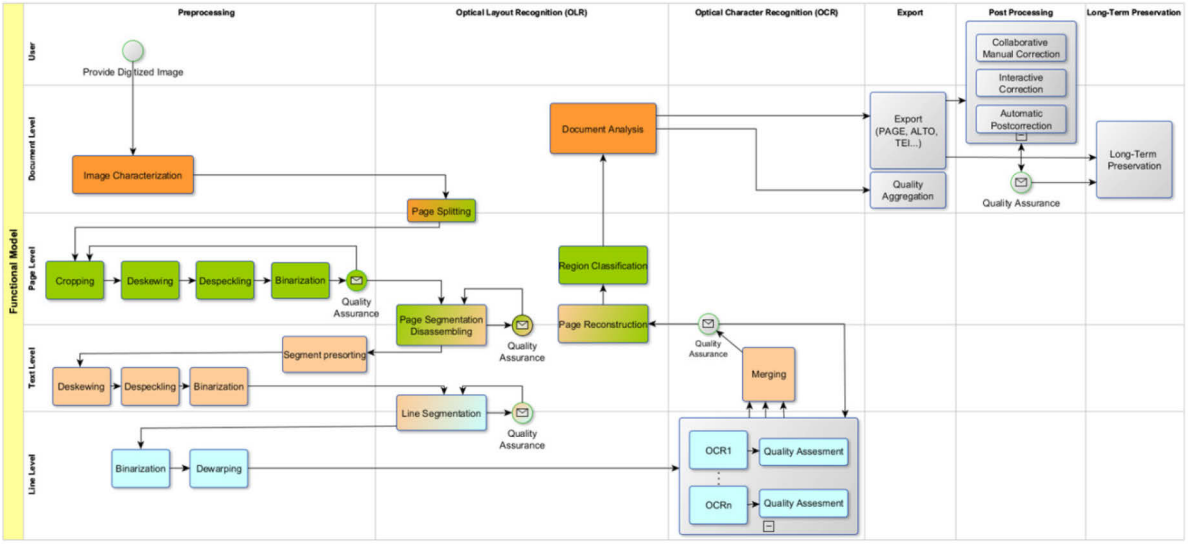

Abb. 1: Konzeptioneller OCR-Workflow in OCR-D

Um die Nutzbarkeit und Interoperabilität der verschiedenen, von den Modulprojektpartnern entwickelten OCR-Komponenten zu gewährleisten, hat das Koordinationsprojekt auf Grundlage etablierter Standards Spezifikationen definiert[3] und mit dem OCR-D/core Framework[4] eine Referenzimplementierung in Python zur Verfügung gestellt.

3 Das OCR-D-Framework

3.1 Spezifikationen und Implementierung

OCR-D nutzt das METS Metadatenformat als zentralen Container zur Prozesskommunikation und Dokumentation. METS ist ein langjähriger und international etablierter Standard für die Kodierung

[3] Github. OCR-D Spezifikationen https://github.com/OCR-D/spec (Letzter Zugriff auf alle Links am zugegriffen am 05.05.2020).

[4] GitHub. OCR-D https://github.com/OCR-D/core.

von deskriptiven, administrativen, technischen und strukturellen Metadaten digitaler Objekte. Standardisierte METS-Dateien werden bei allen DFG-geförderten Digitalisierungsprojekten für jedes digitale Objekt erstellt. Mit dem DFG-Viewer und der METS-Datei können die digitalisierten Bestände online präsentiert werden.[5] Daher setzt OCR-D ebenfalls auf METS auf und reichert dieses im Prozess der Volltextdigitalisierung mit weiteren Informationen der OCR sowie Provenienzinformationen an.

Der Volltext selbst wird in OCR-D im PAGE-Format kodiert. Dieses Format ist XML-basiert und kann sowohl die Text- als auch die Strukturinformationen speichern. Das Format wird als Standard in ICDAR Competitions[6] verwendet. Im Unterschied zum ALTO-Format besitzt dieses Format den Vorteil, dass mehr Informationen über den Text kodiert werden können. Da sich ALTO als Standard in Bibliotheken etabliert hat, wurde von der DHd AG OCR – unter Koordination des OCR-D-Projekts und in Kooperation mit den PAGE-XML Maintainern PRIMA Research Lab – eine PAGE-ALTO-Konversion erarbeitet.

Alle Softwaremodule, die im weiteren Kontext von OCR-D entwickelt wurden, müssen in einer JSON-Datei gemäß dem ocrd-tool JSON Schema[7] beschrieben werden. Darin enthalten sind Informationen über die erforderlichen und optionalen Parameter für eine Software sowie Versionsnummer, verwendetes Neuronales Netz, Verortung im Funktionsmodell usw. Alle OCR-D-Werkzeuge werden daneben als Docker-Container verfügbar gemacht. Dadurch wird deren Installation und Bereitstellung in verschiedenen IT-Umgebungen vereinfacht.[8]

Aus diesen Softwaremodulen kann ein OCR-Workflow, angefangen bei der digitalisierten Seite bis zum transkribierten Text, erstellt werden. Eine Hilfestellung dafür bietet die OCR-D-Workflow-Dokumentation, in der alle möglichen Prozessierungsschritte beschrieben werden.[9] Für jeden Prozessierungsschritt werden zudem die verfügbaren Softwaremodule mit ihren Parametern aufgeführt. Die Dokumentation bietet darüber hinaus Empfehlungen für initiale OCR-Workflows (abhängig von den vorhandenen Ressourcen), die für gängige Vorlagen/Dokumente gute Ergebnisse liefern sollten. Im Einzelfall muss der OCR-Workflow jedoch abhängig von der Vorlage/dem Dokument nachjustiert werden.

[5] METS-Anwendungsprofil für digitalisierte Medien (Version 2.3.1) https://dfg-viewer.de/fileadmin/groups/dfgviewer/METS-Anwendungsprofil_2.3.1.pdf.

[6] Im Rahmen der International Conference on Document Analysis and Recognition (ICDAR) finden jährlich Wettbewerbe statt, bei denen die Teilnehmer Algorithmen für spezifische Problemstellungen bzw. die zur Verfügung gestellten Korpora entwickeln. Vgl. z.B. die Wettbewerbe der ICDAR 2019, https://icdar2019.org/competitions-2/.

[7] https://ocr-d.de/en/spec/ocrd_tool.

[8] https://hub.docker.com/u/ocrd.

[9] https://ocr-d.de/en/workflows.

Am Ende des OCR-Workflows können die generierten Daten inklusive aller Zwischenergebnisse im OCR-D-ZIP-Format[10] gepackt werden. Das Format dient dem standardisierten Datenaustausch zwischen OCR-Produktion und Datenspeichern, wie Repositorium oder Archivspeicher, und baut auf dem BagIt-Standard[11] auf.

Für den kompletten Workflow stehen verschiedene Möglichkeiten zur Verfügung. Ein einfaches Werkzeug für serielle Workflows inklusive Validierung der Ein- und Ausgabedaten ist in der OCR-D core Referenzimplementierung enthalten. Für komplexere Workflows können die makefile-basierte workflow-configuration[12] oder auch der Taverna-Workflow[13] genutzt werden. Letzterer enthält die bereits erwähnten initialen OCR-Workflows für verschiedene Anwendungsfälle (langsamer/schneller Prozessor). Während des OCR-Workflows werden alle Prozessierungsschritte nacheinander ausgeführt und die Provenienzinformationen als PROV-XML[14] gespeichert. Am Ende werden alle erzeugten Dateien im OCRD-ZIP-Format gepackt und können im Forschungsdatenrepositorium[15] gespeichert werden. Dort kann nach Dokumenten gesucht, bzw. nach verschiedenen Kriterien gefiltert werden. Es ist technisch identisch mit dem öffentlich verfügbaren OCR-D-GT-Repositorium.

Das komplette OCR-D-Framework[16] mit allen Softwaremodulen, dem Taverna-Workflow und dem Forschungsdatenrepositorium kann als Docker-Container ohne weitere Anforderungen lokal installiert werden.

3.2 Ground Truth

Eine weitere zentrale Aufgabe innerhalb von OCR-D ist es, ein umfassendes Ground-Truth-Korpus[17] aufzubauen, das Referenz- und Trainingsdaten enthält. Mit den zugehörigen Transkriptionsrichtlinien wird das Korpus zum einen dokumentiert; zum anderen bieten diese Richtlinien ein Instrument, das Korpus in seinem Umfang kontinuierlich erweitern zu können. Die OCR-D-Ground-Truth kann über das OCR-D-GT-Repositorium[18] bezogen werden.

Annotationstiefe

Das Ground-Truth-Korpus bietet Daten in drei Annotationstiefen an:

[10] https://ocr-d.de/en/spec/ocrd_zip.

[11] W3C. BagIt Profiles Specification 1.3.0: https://bagit-profiles.github.io/bagit-profiles-specification/.

[12] https://github.com/bertsky/workflow-configuration.

[13] https://github.com/OCR-D/taverna_workflow.

[14] https://www.w3.org/TR/prov-xml/.

[15] https://github.com/OCR-D/repository_metastore.

[16] https://github.com/VolkerHartmann/ocrd_framework.

[17] Ground Truth sind fehlerfreie, manuell korrigierte Text- und Strukturdaten, die sowohl für die Bewertung als auch für das Training von Softwaremodellen für Text- und Layouterkennung verwendet werden.

[18] https://ocr-d-repo.scc.kit.edu/api/v1/metastore/bagit.

- Strukturregionen, Textzeilen, Wortkoordinaten
- Strukturregionen, Textzeilen
- Textzeilen

Die Erschließungstiefe wird durch spezifische Metadaten[19] dokumentiert, die sich im METS-Metadatensatz befinden. Neben den Strukturierungsinformationen enthalten die Metadaten auch Informationen über spezifische Artefakte der digitalen Vorlage, wie beispielsweise Beschädigungen oder Stempel.

Der Erschließung nach bildet das Korpus zwei große Bestände, deren derzeitigen Umfang die folgende Tabelle zeigt. Die Daten des Korpus werden in PAGE-XML[20] gespeichert.

Tab. 1:

Text-Struktur-Korpus	Struktur-Korpus
16000 Zeilen/2.814 Regionen	8400 Regionen

Die Arbeiten zur Erstellung, Pflege und Verbesserung des Ground Truth werden vom Koordinationsgremium als eine kontinuierliche, kollaborative Aufgabe gesehen.

3.3 Modulprojekte

Insgesamt acht Modulprojekte arbeiteten in der zweiten Förderphase des OCR-D Projektes an der Entwicklung von Werkzeugen (im weiteren als *Prozessoren* bezeichnet). In den folgenden Abschnitten stellen die Modulprojekte die wichtigsten Ergebnisse ihrer Software-Entwicklung vor.

3.3.1 Skalierbare Verfahren der Text- und Strukturerkennung für die Volltextdigitalisierung historischer Drucke[21]

Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)

Das DFKI war als Projektpartner im OCR-D Projekt mit zwei Modulen vertreten: Bildoptimierung und Layouterkennung. In beiden Modulen wurden mehrere Prozessoren entwickelt und in das OCR-D-Softwaresystem integriert.

Das erste Modul-Projekt *Bildoptimierung* fokussierte sich auf die Vorverarbeitung der Digitalisate mit dem Ziel, die Bildqualität und somit auch die Performanz der nachfolgenden OCR-Module zu

[19] Github. gt-labelling: Labeling OCR ground truth data with special METS-Profile: https://github.com/OCR-D/gt-labelling.

[20] Primaresearch. PAGE XML Format for Page Content: https://www.primaresearch.org/schema/PAGE/gts/pagecontent/2019-07-15/Simple%20PAGE%20XML%20Example.pdf.

[21] Verfasst von Prof. Dr. Prof. h.c. Andreas Dengel und Martin Jenckel.

verbessern. Dafür wurden Werkzeuge für die Binarisierung, das Deskewing, das Cropping und das Dewarping implementiert.

Das Cropping-Werkzeug ist als besonders performant hervorzuheben. Auch das Dewarping-Werkzeug ist aufgrund seiner neuartigen Architektur interessant. Mit Hilfe generativer neuronaler Netze werden entzerrte Varianten von Bildern generiert, anstatt explizite Transformationen für die Entzerrung zu bestimmen. Tests zeigen, dass 84,96% der Pixel korrekt klassifiziert werden.

Im zweiten Modul-Projekt *Layouterkennung* galt es, die Dokumentstruktur, sowohl einzelner Dokumentseiten als auch des Gesamtdokuments, zu extrahieren. Die meisten OCR-Methoden können z.B. nur einzelne Textzeilen verarbeiten. Die entwickelten Werkzeuge dienen der Text-Nicht-Text-Segmentierung, der Regionensegmentierung und -klassifizierung, der Textzeilenerkennung sowie der Strukturanalyse auf Dokumentebene.

Ein Entwicklungsschwerpunkt war die kombinierte Regionensegmentierung und -klassifizierung, welche auf der, aus der Video- und Bildsegmentierung bekannten, MaskRCNN-Architektur basiert. Dieses Werkzeug nutzt die unbearbeiteten Rohdaten, sodass einerseits keine Vorverarbeitung (wie bspw. Binarisierung) notwendig ist und andererseits das volle Informationsspektrum ausgenutzt werden kann.

3.3.2 Weiterentwicklung eines semiautomatischen Open Source Tools zur Layout-Analyse und Regionen-Extraktion und -klassifikation (LAREX) von frühen Buchdrucken[22]

Lehrstuhl für Informatik VI, Julius-Maximilians-Universität Würzburg

Das primäre Ziel dieses Modulprojekts ist die automatische Segmentierung von Scans frühneuzeitlicher Buchdrucke. Dazu wurde ein sogenannter Pixel-Classifier entwickelt, ein Fully Convolutional Neural Net, das jedem Pixel eine in den Trainingsdaten vordefinierte Kategorie zuordnet und auf einer Weiterentwicklung der U-Net Architektur basiert (vgl. Abb. 2).[23]

[22] Verfasst von Alexander Gehrke und Prof. Dr. Frank Puppe.

[23] Vgl. Wick und Puppe (2018).

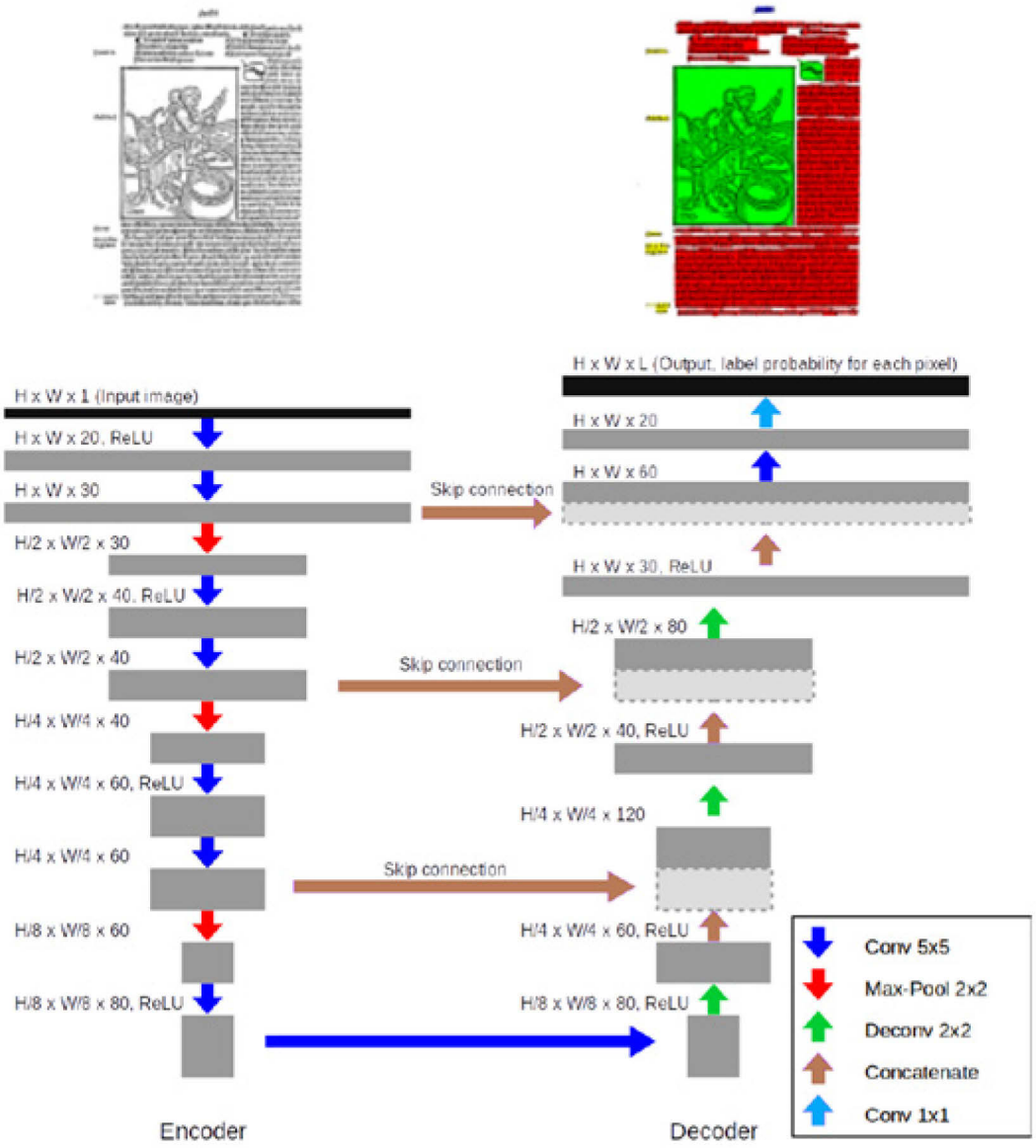

Abb. 2: Architektur des verwendeten Pixel-Classifiers zur Segmentierung. Die beiden Bilder oben zeigen den Input (links) und den Output mit segmentierten Regionen (rechts), die durch verschiedene Farben visualisiert sind. Aus dem Binärbild einer Buchseite wird über viele Schichten in einer Encoder/Decoder-Struktur mit Skip-Connections die Zuordnung von Pixeln zu Regionen erzeugt, wobei die Regionentypen variabel trainierbar sind.

Der Pixel-Classifier benötigt als tiefes neuronales Netz mit vielen Schichten umfangreiche Trainingsdaten (Ground Truth). Die im Projektverlauf zur Verfügung stehenden, annotierten Datensätze wie zum Beispiel vom Deutschen Textarchiv waren für den Einsatzzweck nicht optimal geeignet, da es nicht ausreicht, eng benachbarte Regionen in alten Drucken nur mit Rechtecken auszuzeichnen. Die mit diesen Datensätzen trainierten Modelle für die Neuronalen Netze zeigten relativ schlechte Segmentierungsergebnisse. Daher wurde der in den Vorarbeiten erstellte,

semiautomatische LAREX-Editor[24] weiterentwickelt und in den interaktiven OCR-Workflow OCR4all eingebettet,[25] sodass in Kooperation mit Partnern präzise und umfangreiche Ground Truth in effizienter Weise erzeugt werden konnte (Zeitaufwand ca. eine halbe bzw. eine Minute für die semi-automatische Segmentierung einer Scan-Seite von anspruchsvollen Drucken aus dem 15. und 16. Jahrhundert für erfahrene bzw. unerfahrene Nutzer).

Die anhand der neuen Ground Truth trainierten Modelle wurden mit einem repräsentativen Testdatensatz evaluiert (171 repräsentative Scan-Seiten aus 54 unterschiedlichen Büchern des 16. bis 19. Jahrhunderts, der vom OCR-D-Team bereitgestellt und zur Evaluation geringfügig nachkorrigiert wurde). Damit wurde ein pixelbasierter F1-Score für Textregionen von ca. 99,4% und für die schwierigeren Bildregionen von ca. 91,4% erzielt, wenn der Seitenrand in der Vorverarbeitung bereits entfernt wurde. Ohne Seitenrandentfernung sinken die Werte auf ca. 99% und 83%, was den Nutzen der Vorverarbeitung im OCR-Workflow verdeutlicht. Weitere Verbesserungen nach Projektabschluss könnten erreicht werden, indem noch mehr Ground Truth annotiert, durch Techniken wie Voting effizienter genutzt sowie alternative Ansätze erprobt werden, die strukturelle Elemente wie Konturen oder Baselines von Buchstaben und Texten auswerten.

3.3.3 Optimierter Einsatz von OCR-Verfahren – Tesseract als Komponente im OCR-D-Workflow[26]

Universität und Universitätsbibliothek Mannheim

Im Fokus des Modulprojekts stand die OCR-Software Tesseract, die seit 1985 von Ray Smith entwickelt wurde, seit 2005 als Open Source unter einer freien Lizenz. Das Projekt umfasste zwei Hauptziele: Die Einbindung von Tesseract in den OCR-D-Workflow inklusive Unterstützung der anderen Modulprojekte durch die Bereitstellung von Schnittstellen, sowie die allgemeine Verbesserung der Stabilität, Codequalität und Performance von Tesseract.

Die Einbindung in den OCR-D-Workflow erforderte wesentlich weniger Aufwand als ursprünglich geplant; hauptsächlich, weil die meiste Arbeit bereits außerhalb des Modulprojekts geleistet worden war und dabei die schon vorhandene Python-Schnittstelle tesserocr[27] genutzt werden konnte.
Für das OCR-D-Modulprojekt der Universität Leipzig wurde Tesseract um die Generierung von alternativen OCR-Ergebnissen für die Einzelzeichen erweitert. Als Eingabedaten für ein OCR-

[24] Vgl. Reul et al (2017).

[25] Universität Würzburg: https://www.uni-wuerzburg.de/de/zpd/ocr4all/.

[26] Verfasst von Stefan Weil.

[27] https://github.com/sirfz/tesserocr.

Postkorrektur-Modell lässt sich damit die Texterkennung weiter verbessern. Ein wertvoller Nebeneffekt des neuen Codes sind genauere Zeichen- und Wortkoordinaten.

Mit mehreren hundert Korrekturen konnte die Codequalität signifikant gesteigert und ein deutlich stabilerer Programmfluss erreicht werden. Tesseract ist jetzt wartbarer, u.a. dank der verstärkten Nutzung von Standardfunktionen und -datentypen anstelle von proprietärem Code. Der Testcode für unit tests umfasste im Februar 2018 lediglich 546 Codezeilen, inzwischen – zwei Jahre später – sind es 12.277 Codezeilen. Diese Tests decken wesentlich mehr Funktionalität von Tesseract ab und helfen so, die erreichte Qualität zu halten. Überflüssige Teile der Programmierschnittstelle (API) wurden entfernt, der entsprechende Code von 21.378 auf 5.463 Zeilen verkleinert. Die statische Speichergröße für das Tesseract-Programm reduzierte sich im gleichen Zeitraum von 5,2 MB auf 2,8 MB trotz erweiterter Funktionalität; hauptsächlich durch die Entfernung redundanten Codes und größerer statischer Arrays, die nur in bestimmten Nutzungsszenarien benötigt werden. Auch der dynamische Speicherbedarf konnte durch optimierte Datenstrukturen reduziert werden. Tesseract ist signifikant schneller geworden. Beispielsweise benötigen 25 typische Seiten der *Weisthümer* von Jacob Grimm mit Standardparametern auf einem modernen Büro-PC statt 85 Sekunden nur noch 74 Sekunden. Verzichtet man auf die Erkennung von invertiertem Text, der nur in speziellen Drucken vorkommt, reduziert sich der Zeitbedarf weiter auf 60 Sekunden.

Eine wesentliche Verbesserung der Erkennungsgenauigkeit für den Großteil der OCR-D-relevanten Druckwerke konnte durch neue generische Modelle für Tesseract erreicht werden. Diese wurden ab September 2019 auf Basis der Datensammlung *GT4HistOCR*[28] trainiert. Die für historische Drucke passenden Standardmodelle von Tesseract (frk und Fraktur) haben systematische Schwächen wie z.B. das fehlende Paragraphenzeichen oder die falsche Erkennung der Ligaturen *ch* und *ck*, die auf unvollständige und fehlerhafte Trainingsdaten zurückzuführen sind. Mit den neuen Modellen sind diese Schwächen eliminiert und für ein breites Spektrum von Drucken können gute OCR-Ergebnisse erzielt werden. Typische Erkennungsraten liegen deutlich über 90% für Zeichen, die Zeichenfehlerrate (CER) also unter 10%. Wie bei anderer OCR-Software auf Basis neuronaler Netze lässt sich diese Fehlerrate durch ein einfaches werkspezifisches Nachtraining auf unter 1% reduzieren.

[28] https://zenodo.org/record/1344132.

3.3.4 NN/FST – Unsupervised OCR-Postcorrection based on Neural Networks and Finite-state Transducers[29]

Institut für Informatik: Abteilung Automatische Sprachverarbeitung, Universität Leipzig

Eine vollautomatische Nachkorrektur separat von der eigentlichen OCR ist immer nur dann sinnvoll, wenn dabei statistisches Wissen über "richtigen Text" und über typische OCR-Fehler *a priori* hinzukommt. Dafür eignen sich neuronale Netze (NN) ebenso wie gewichtete endliche Transduktoren (WFST), die auf entsprechenden zusätzlichen Daten trainiert werden können.

Für die Umsetzung einer kombinierten Architektur aus NN und FST wurde entschieden, drei Module zu implementieren:

1. eine reine NN-Lösung mit durchgehend (*end-to-end*) trainiertem Modell allein auf Zeichenebene – als tiefes (mehrschichtiges), bidirektionales rekurrentes Netzwerk nach dem Encoder-Decoder-Schema (für verschiedene Eingabe- und Ausgabelänge) mit Attention-Mechanismus und A*-Beamsearch mit einstellbarer Rückweisungsschwelle (gegen Überkorrektur), d.h. die Nachkorrektur von Textzeilen wird wie maschinelle Übersetzung behandelt,[30]
2. ein NN-Sprachmodell (LM) auf Zeichenebene – als tiefes (mehrschichtiges), bidirektionales rekurrentes Netzwerk mit Schnittstelle für Graph-Eingabe und inkrementeller Dekodierung,[31]
3. eine WFST-Komponente mit explizit zu trainierendem Fehlermodell auf Zeichenebene und Wortmodell/Lexikon, sowie Anbindung an 2. – per WFST-Komposition von Eingabegraph mit Fehler- und Wortmodell nach Sliding-Window-Prinzip, Konversion der Einzelfenster zu einem Hypothesengraph pro Textzeile, und Kombination der jeweiligen Ausgabegewichte mit LM-Bewertungen in einer effizienten Suche nach dem besten Pfad.[32]

Die Kombination von 3. mit 2. stellt also eine hybride Lösung dar. Aber auch 1. kann von 2. profitieren (sofern die gleiche Netzwerk-Topologie benutzt wird), indem die Gewichte aus einem auf größeren Mengen reinen Text trainierten Sprachmodell initialisiert werden (Transfer-Learning).

Beide Ansätze profitieren von einer engen Anbindung an den OCR-Suchraum, d.h. eine Übergabe alternativer Zeichen-Hypothesen und ihrer Konfidenz (wie bisher nur mit Tesseract möglich und in Zusammenarbeit mit dem Modulprojekt der UB Mannheim realisiert). Sie liefern aber auch auf

[29] Verfasst von Robert Sachunsky, Lena K. Schiffer, Dr. Maciej Janicki und Prof. Dr. Gerhard Heyer.

[30] GitHub. cor-asv-ann: https://github.com/ASVLeipzig/cor-asv-ann.

[31] GitHub. ocrd_keraslm: https://github.com/OCR-D/ocrd_keraslm.

[32] Github. cor-asv-fst: https://github.com/ASVLeipzig/cor-asv-fst.

reinem Volltext bereits gute Ergebnisse (mit CER-Reduktion von bis zu 5%), sofern genügend passende Trainingsdaten zur Verfügung stehen und die OCR ihrerseits brauchbare Ergebnisse (unterhalb 10% CER) liefert.

Für alle Module stehen Kommandozeilen-Schnittstellen für Training und Evaluierung, sowie volle OCR-D-Schnittstellen für Prozessierung und Evaluierung zur Verfügung.[33]

3.3.5 Automatische Nachkorrektur historischer OCR-erfasster Drucke mit integrierter optionaler interaktiver Korrektur[34]

Centrum für Informations- und Sprachverarbeitung (CIS), Ludwig-Maximilians-Universität München

Das Ergebnis des Projekts ist ein in den OCR-D-Workflow integriertes System *A-I-PoCoTo* zur vollautomatischen Nachkorrektur OCR-erfasster historischer Drucke. Das System beinhaltet zudem eine optional nachgeschaltete interaktive Nachkorrektur (*I-PoCoTo*), die in das interaktive Nachkorrektursystem *PoCoWeb* eingebunden ist. Das System kann damit auch alternativ als Stand-Alone-Tool zur gemeinschaftlichen webbasierten Nachkorrektur von OCR-Dokumenten eingesetzt werden.

Die Grundlage der vollautomatischen Nachkorrektur ist ein flexibles, featurebasiertes Machine-Learning (ML) Verfahren zur vollautomatischen OCR-Nachkorrektur mit einem besonderen Fokus auf die Vermeidung der Verschlimmbesserungsproblematik. Zur Erkennung von Fehlern und für die Erzeugung von Korrekturkandidaten verwendet das System die am CIS entwickelte dokumentenabhängige Profilierungstechnologie.[35] Die Features des Systems verwenden neben verschiedenen Konfidenzwerten insbesondere auch Informationen aus zusätzlichen OCR-Engines.

Zur Auswertung der vollautomatischen Nachkorrektur wurden einerseits Bodensteins „Wie Sich Meniglich“ (1557)[36] und ein Ausschnitt von „Die Grenzboten“ (1847)[37] verwendet, wobei jeweils eine Hälfte zum Training, die andere für die Auswertung eingesetzt wurde. Die folgende Abbildung zeigt jeweils die wortweise Genauigkeit der OCR-Erkennung (blau) sowie die wortweise Genauigkeit nach der vollautomatischen Nachkorrektur (orange).

[33] Zu den Entwicklungen dieses Modulprojekts vgl. ausführlicher Sachunsky et al. (2012).

[34] Verfasst von Dr. Florian Fink und Prof. Dr. Klaus U. Schulz.

[35] Github. The CIS language aware OCR document error profiler: https://github.com/cisocrgroup/Profiler.

[36] Zenodo. GT4HistOCR: https://zenodo.org/record/1344132.

[37] Github. Resources: https://github.com/cisocrgroup/Resources.

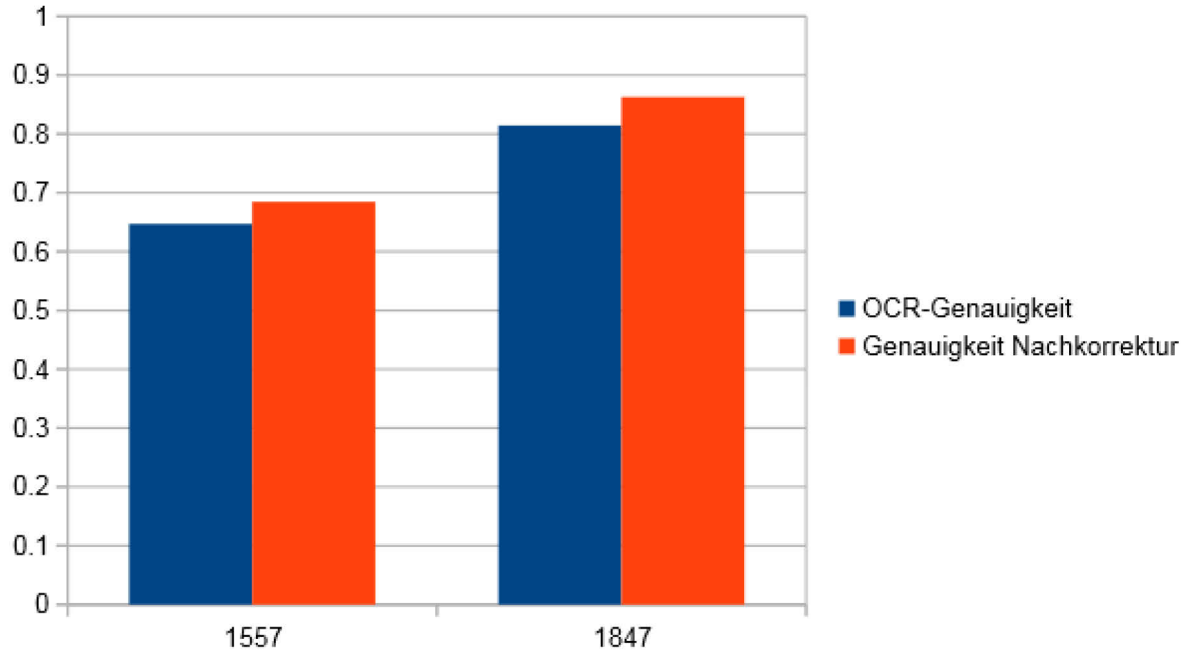

Abb. 3: Verbesserung der Erkennungsrate durch die Nachkorrektur am Beispiel zweier Drucke

Das System protokolliert sämtliche Korrekturentscheidungen. Über diesen Protokollmechanismus kann die automatische Postkorrektur in *PoCoWeb* interaktiv überprüft werden. Dabei können sowohl einzelne getätigte Korrekturentscheidungen manuell rückgängig gemacht werden, als auch nicht getätigte Korrekturentscheidungen nachträglich ausgeführt werden.

Das gesamte System ist in den OCR-D-Workflow eingebunden und folgt den dort gültigen Konventionen.

3.3.6 Entwicklung eines Modellrepositoriums und einer automatischen Schriftarterkennung für OCR-D[38]

Lehrstuhl für Digital Humanities, Universität Leipzig; Lehrstuhl für Informatik 5 (Mustererkennung), Friedrich-Alexander-Universität Erlangen-Nürnberg; Gutenberg-Institut für Weltliteratur und schriftorientierte Medien (Abteilung Buchwissenschaft), Johannes Gutenberg-Universität Mainz

Die Erkennungsquoten von OCR für Drucke, die vor 1800 produziert wurden, variieren sehr stark, da die Diversität historischer Schriftarten in den Trainingsdaten entweder gar nicht oder nur unzureichend berücksichtigt wird. Dem begegnet dieses Modulprojekt mithilfe eines Werkzeugs[39] zur automatischen Erkennung von Schriftarten in Bilddigitalisaten, das ein Convolutional Neural

[38] Verfasst von Dr. Nikolaus Weichselbaumer, Dr. Saskia Limbach, Mathias Seuret, Rui Dong, Prof. Dr. Manuel Burghardt und Dr. Vincent Christlein.

[39] Github. ocrd_typegroups_classifier: https://github.com/OCR-D/ocrd_typegroups_classifier.

Network, konkret ein DenseNet-121[40] einsetzt. Das Netzwerk wurde mit 35.000 manuell annotierten Bildern trainiert und erreicht eine Genauigkeit von 98% bei der Bestimmung von Schriftarten.[41] Dabei unterscheidet es zwischen Antiqua, Kursiv, Textura, Rotunda, Gotico-Antiqua, Bastarda, Fraktur, Schwabacher, Griechisch und Hebräisch sowie leeren Seiten und solchen mit Artefakten, die nicht gedruckte Schrift sind (Holzschnitte, Kupferstiche, Handschrift etc.).

In einem zweiten Schritt wurde die Online-Trainingsinfrastruktur Okralact erstellt.[42] Sie ermöglicht es, über eine einheitliche Schnittstelle unterschiedliche OCR-Engines (Tesseract, Ocropus, Kraken, Calamari) zu trainieren.[43] Die mit Okralact trainierten Modelle können anschließend im frei verfügbaren Repositorium Zenodo archiviert und der Community zur Verfügung gestellt werden. Um hier einen Grundstock zu legen, wurden ca. 2.500 Zeilen für Bastarda, Textura und Schwabacher aus einer Vielzahl unterschiedlicher Bücher transkribiert.

Die hohe Genauigkeit des Werkzeugs zur Erkennung von Schriftarten eröffnet die Möglichkeit, das Werkzeug in Zukunft durch weitere Trainingsdaten sogar zwischen den Schriften einzelner Drucker unterscheiden zu lassen, was mehrere Desiderate von Bibliotheken und historischer Forschung adressieren würde.

3.3.7 OLA-HD – Ein OCR-D Langzeitarchiv für historische Drucke[44]

Georg-August-Universität Göttingen; Niedersächsische Staats- und Universitätsbibliothek; Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen

Die massenhafte Digitalisierung von historischen Drucken wurde in den letzten Jahren stark vorangetrieben. Daher stehen mittlerweile zahlreiche Digitalisate zur Verfügung, für deren weitere Nutzung eine nachhaltige Archivierung und Identifizierung der Digitalisate, der bibliographischen Metadaten sowie der erschlossenen Volltexte und deren Versionen notwendig ist.

Ziel des OCR-D-Modulprojektes *OLA-HD – Ein OCR-D Langzeitarchiv für historische Drucke* ist die Entwicklung eines integrierten Konzepts für die Langzeitarchivierung, Versionierung und die persistente Identifizierung von OCR-Objekten sowie eine prototypische Implementierung**.**

Im regelmäßigen Austausch mit den Projektpartnern wurden die Basisanforderungen für die Langzeitarchivierung und persistente Identifikation ermittelt und in Form einer Spezifikation zur technischen und wirtschaftlich-organisatorischen Umsetzung festgehalten.

[40] Vgl. Huang et al (2017).

[41] Vgl. Seuret et al. (2019).

[42] https://github.com/OCR-D/okralact.

[43] Vgl. Baierer et al. (2019).

[44] Verfasst von Triet Ho Anh Doan, Zeki Mustafa Doğan, Jörg-Holger Panzer, Kristine Schima-Voigt und Prof. Dr. Philipp Wieder.

Mit dem OLA-HD-Prototypen[45] kann der Anwender OCR-Ergebnisse eines Werkes als komprimiertes Dateiformat (OCRD-ZIP[46]) in das System laden. Das System validiert die Datei, vergibt eine PID und schickt die Datei an den Archiv-Manager (CDSTAR – GWDG Common Data Storage Architecture[47]). Dieser schreibt die Datei in das Archiv (Bandspeicher) und indexiert die Daten für die Suche. Abhängig von der Konfiguration (Datei-Typ, Datei-Größe etc.) werden Dateien zusätzlich in einem Online-Speicher (Festplatte) gespeichert, um einen schnellen Zugriff zu ermöglichen. Der Nutzer hat dabei Zugriff auf alle OCR-Versionen und kann diese zudem als OCRD-ZIP Dateien herunterladen. Alle Werke und Versionen haben eigene PIDs. Die PIDs werden vom European Persistent Identifier Consortium (ePIC)[48] Service generiert. Die verschiedenen OCR-Versionen eines Werkes sind über die PID verknüpft, so dass das Datenmanagementsystem die Versionierung in einer Baumstruktur abbilden kann.

Abgesehen von dem Hochladen von Daten und dem Herunterladen von hochaufgelösten Bildern können Gastbenutzer das System in vollem Umfang nutzen. Sie können beispielsweise das Repositorium durchsuchen und eine Vorschau von Text und – falls verfügbar – Bildern in der Dateistruktur erhalten oder durch die verschiedenen Versionen navigieren. Nutzer können sich registrieren und ihre Daten in einem Dashboard verwalten.

Neben dem Prototypen wird ein Konzept zu dessen Umwandlung in ein Produkt erstellt. In einer möglichen weiteren Phase ist es vorgesehen, eine Integration von OLA-HD in den Digitalisierungsworkflow zu ermöglichen, beispielsweise durch die Implementierung eines Workflowschrittes in die Workflow Management Software Goobi bzw. Kitodo,[49] die Integration in das VD18-Portal[50] sowie in den OCR-Workflow selbst.

3.4 Die OCR-D-Software im Gesamten

Der offene Quellcode und der modulare Aufbau der Software erlauben es, für die einzelnen zu prozessierenden Bilddigitalisate die jeweils am besten für diese Aufgabe geeigneten Prozessoren auszuwählen und zu einem optimalen OCR-Workflow zusammenzustellen. Die Prozessoren können sowohl einzeln, bspw. zu Testzwecken, oder auch gesammelt aufgerufen werden. Zwei Software-Lösungen zur Spezifikation und Ausführung von Workflows (*Taverna* und *workflow-config*) enthalten

[45] https://github.com/subugoe/OLA-HD-IMPL.
[46] https://ocr-d.de/de/spec/ocrd_zip.
[47] https://cdstar.gwdg.de/.
[48] https://www.pidconsortium.net/.
[49] https://www.sub.uni-goettingen.de/digitale-bibliothek/digitale-werkzeuge/goobi-digital-library-modules/.
[50] https://gso.gbv.de/DB=1.65/.

jeweils mehrere erfolgreich getestete Standard-Workflows, die als Ausgangspunkt für die Spezifikation eines, für ein bestimmtes Bilddigitalisat, optimalen Workflows dienen können.

4 Ergebnisse der OCR-D-Teststellung

4.1 Hintergrund

Um die Jahreswende 2019/2020 wurde die OCR-D-Software in neun Pilotbibliotheken auf ihre Funktionalität und Einsetzbarkeit in der (bibliothekarischen) Praxis getestet. Daran teilgenommen haben neben den Häusern des Koordinierungsprojekts auch zwei an den Modulprojekten beteiligte Bibliotheken sowie vier weitere Bibliotheken. Die Erkenntnisse dieses ersten Testlaufs dienen der Weiterentwicklung des OCR-D-Prototypen, dessen Akzeptanz bei den künftigen Nutzern sichergestellt werden soll.

Alle Pilotbibliotheken haben bereits zuvor, zumindest auf Projektebene bzw. über Dienstleister, mit OCR gearbeitet und erwägen, die Volltexterkennung perspektivisch eigenständig durchzuführen. Von einer OCR-Software wird insbesondere eine hohe Texterkennungsrate erwartet, weiterhin wünschenswert sind hohe Layouterkennungsraten, kostengünstiger Einsatz, schnelle Adaptierbarkeit, Ausgabe von Standardformaten, Anbindung an existierende Workflows und gut dokumentierte Schnittstellen.

4.2 Auswertung der Softwaretests

Um die Rückmeldungen der Pilotbibliotheken miteinander vergleichen zu können, wurde zu Beginn der Teststellung ein einheitlicher Fragebogen ausgegeben. Abgefragt wurden die Rahmenbedingungen des Testlaufs, wie die verwendete technische Ausstattung und die getesteten OCR-D-Prozessoren, die Dokumentation der Software, Schnittstellen, Funktionalität bzw. Benutzbarkeit der Software, deren Einbindbarkeit in existierende Workflows und die benötigten Ausgabeformate. Zudem wurden mit Erkennungsqualität, Funktionalität bzw. Benutzbarkeit, offenen Anforderungen sowie positiven Merkmalen der OCR-D-Software die Ergebnisse der Tests erhoben.

Für die Teststellung wurde die OCR-D-Software mit und ohne Docker auf einer Reihe unterschiedlich leistungsstarker, teils virtueller Server installiert. Die Installation war auf allen Servern möglich, bei Nicht-Intel-Rechnern allerdings aufwändiger. Insbesondere die neue Gesamt-Installation der verfügbaren Prozessoren (ocrd_all)[51] ist jedoch einfach und schnell durchführbar.

Schwierigkeiten gab es dagegen beim Verständnis des Einsatzbereichs der verschiedenen Prozessoren und insbesondere mit deren Zusammenstellung zu sinnvollen Workflows. Dies war

[51] https://github.com/OCR-D/ocrd_all.

durch die zum Zeitpunkt der Teststellung noch fehlende zentrale Gesamtdokumentation bedingt, in der die Funktionsweise des OCR-D-Frameworks sowie der einzelnen Prozessoren erklärt und mögliche funktionsfähige Workflows vorgestellt werden. Die damalige Dokumentation war noch mehr auf Entwickler bzw. Personen mit Kenntnissen im Bereich OCR ausgerichtet und erforderte von fachfremden Testern eine längere Einarbeitungszeit. Die Vorschläge und Wünsche der Tester zu einer Nutzerdokumentation sind noch in die Neubearbeitung eingeflossen.
Insgesamt läuft die Software sehr stabil, in keiner Bibliothek kam es zu Abbrüchen. Alle gewünschten Ausgabeformate werden unterstützt, die wenigen noch benötigten Schnittstellen sind Teil der geplanten Weiterentwicklung. In Verbindung mit einer Workflow-Software wie bspw. Kitodo wurde die OCR-D-Software aufgrund des engen Zeitrahmens in keiner Bibliothek getestet. Besonders positiv hervorgehoben wurden von den Testern die Modulprojektprozessoren *ocrd-tesserocr-recognize* zur Texterkennung und *ocrd-anybaseocr-crop* zur Bildvorverarbeitung, sowie der ebenfalls für die Vorverarbeitung benötigte OCR-D-kompatible Prozessor *ocrd-olena-binarize*.

Die Erkennungsqualität wurde von den Pilotbibliotheken auf Basis von selbst erstellter Ground Truth stichprobenhaft auf einzelnen Seiten ermittelt. Diese ersten Ergebnisse sind sehr vielversprechend. Je nach verwendeten Prozessoren und prozessiertem Digitalisat, insbesondere Umfang und Art der Störungen (wie z.B. verzerrte und schiefe Zeilen oder stark durchscheinender Widerdruck) sowie der verwendeten Type, werden unterschiedliche Fehlerraten erzielt. Jedoch offenbarten sich bei der Layouterkennung bzw. Segmentierung noch einige Probleme mit der Erkennungsqualität. Es wurde eingeschätzt, dass diese nur manuell korrigiert werden können. Der Umfang der Test-Korpora lässt solch eine Verfahrensweise noch zu, aber die Prozessierung kompletter Bibliotheksbestände setzt eine bessere Layout-Segment-Erkennung voraus.

Mithilfe des von Robert Sachunsky entwickelten Workflow-Werkzeugs OCR-Workflow-configuration wurden bereits vor mehreren Monaten verschiedene Digitalisierungsworkflows auf ihre Erkennungsqualität hin getestet. Damals lag die niedrigste CER noch bei 7,4%.[52] In Zusammenhang mit der Erkennungsqualität wurden am KIT Tests mit 2405 Digitalisierungsworkflows durchgeführt, im Zuge derer die Qualität der Prozessoren über die CER abgeschätzt wird. Als Vorlagen wurden bei diesen neuen Tests verschiedene Dokumente aus dem 17. und 18. Jahrhundert verwendet, die im OCR-D-Ground-Truth-Repository zur Verfügung stehen. Es zeigte sich, dass die Erkennungsqualität nicht nur vom gewählten Digitalisierungsworkflow, sondern wesentlich stärker von den Besonderheiten der Vorlage abhängig ist. Für einfach strukturierte Seiten konnten fast alle Digitalisierungsworkflows gute Ergebnisse erzielen. In der Spitze lag die Fehlerrate bei guten 0,8%

[52] Vgl. https://github.com/bertsky/workflow-configuration.

CER. Bei Vorlagen mit Fußnoten und Initialen kam es jedoch durchgehend zu schlechteren Ergebnissen. Vor allem die Segmentierung bereitete hier Probleme, wohingegen die Schrifterkennung bei korrekter Segmentierung gut funktioniert. Die initialen OCR-Workflows[53] sind dennoch ein guter Ausgangspunkt, um je nach Vorlage einzelne Prozessoren durch geeignetere zu ersetzen oder Parameter feiner zu justieren.

Desiderate der Pilotbibliotheken an die OCR-D-Software betreffen die drei Bereiche Dokumentation, Qualität bzw. Benutzbarkeit der Prozessoren sowie perspektivisch deren Skalierbarkeit. Die Anforderungen an die Dokumentation werden laufend umgesetzt. Bei den Prozessoren sind insbesondere Verbesserungen bei der Layouterkennung wünschenswert, zudem werden noch funktionsfähige Prozessoren für die Nachkorrektur benötigt. Die Prozessoren der Modulprojekte konnten aufgrund ihrer besonderen technischen Anforderungen (GPU) bzw. entwicklungsbedingt noch kaum in die Teststellung einbezogen werden. Deren Ergebnisse werden hoffentlich die oben genannten Desiderate bedienen. Verbesserungen könnten auch durch die Bereitstellung weiterer Modelle erreicht werden. Perspektivisch und mit Blick auf die geplante Volltexttransformation der VD sollten die Laufzeit der Prozessoren (v.a. von Dewarping, Denoising, Clipping und Resegmentation) optimiert und Möglichkeiten zur Parallelisierung geboten werden. Die OCR-D-Software ist zwar auf Skalierbarkeit ausgerichtet, deren Optimierung war jedoch nicht Teil der derzeitigen Projektphase.

Besonders positiv sehen die Tester den modularen Aufbau der OCR-D-Software. Dieser ermöglicht es, aus mehreren Prozessoren den am besten für einen bestimmten Anwendungsfall geeigneten Workflow zusammenzustellen. Zudem können so die einzelnen Prozessoren von verschiedenen Expertengruppen (weiter-)entwickelt werden. Experimente zu einzelnen Schritten im OCR-Workflow werden erst durch diese Modularität ohne aufwändige Programmierarbeiten möglich. Darüber hinaus ist die Prozessierung mit OCR-D letztlich relativ einfach anzustoßen und bei Bedarf erhält der Anwender schnell und unkompliziert Support durch die Entwickler.[54] Die Software läuft insgesamt sehr stabil und deren Open-Source-Politik wird begrüßt.

5 Ergebnisse der Umfrage mit VD-Bibliotheken

5.1 Hintergrund

Mit Blick auf den Einsatz der OCR-D-Software in bestandshaltenden Einrichtungen zur Volltexttransformation der in VD16, 17 und 18 erfassten Titel wurde um die Jahreswende 2019/2020

[53] https://ocr-d.de/en/workflows.

[54] Für schnelle Rückmeldungen eignet sich insbesondere der OCR-D Gitter chatroom: https://gitter.im/OCR-D/Lobby.

mit den Bibliotheken, die an den VD beteiligt sind, eine Umfrage zum derzeitigen Stand der Digitalisierung in ihren Häusern und den VD durchgeführt. An der Umfrage teilgenommen haben alle vier Trägerbibliotheken der VD sowie 14 der 41 weiteren, über die Mailingliste des VD17 angeschriebenen Bibliotheken. Es handelt sich dabei v.a. um die im Bereich der Digitalisierung sehr stark bis mittel engagierten Bibliotheken. Die Umfrage kann daher die Erfahrungen, Einschätzungen und Planungen der VD-Bibliotheken darstellen, die in großem oder mittlerem Umfang an den Digitalisierungsarbeiten beteiligt sind bzw. waren. Anzunehmen ist, dass sich vornehmlich diese Bibliotheken auch in eine Volltexttransformation der VD einbringen würden.

5.2 Digitalisierung in den VD-Bibliotheken

Bis auf eine sehr kleine Bibliothek werden in allen befragten Einrichtungen mehrere tausend Seiten pro Monat von mehreren Stellen mit einer Workflow-Software bilddigitalisiert. Überwiegend kommen dabei Kitodo und Goobi zum Einsatz.

Obwohl in den befragten Bibliotheken zahlreiche Digitalisierungsprojekte unternommen werden bzw. sich derzeit in Planung befinden, erwähnt lediglich ein Drittel der Umfrageteilnehmer aktuelle Projekte mit Volltexterkennung. 61% der Bibliotheken haben jedoch zumindest in zwei Projekten bereits erste Erfahrungen mit OCR gesammelt. Von diesen führt die Hälfte die Texterkennung selbst durch, während 18% dafür einen Dienstleister einsetzen. Die übrigen Bibliotheken haben bereits beide Varianten ausprobiert. Zum Einsatz kommt hauptsächlich die Software ABBYY Finereader, daneben auch Tesseract. Für die Erstellung der OCR durch einen Dienstleister sprechen aus Sicht der Umfrageteilnehmer der verminderte organisatorische Aufwand sowie fehlende OCR-Kenntnisse in den Einrichtungen.

5.3 OCR-Projekte mit VD-Titeln

Aufgeschlossen gegenüber einem OCR-Projekt zu VD-Titeln an ihrer Bibliothek sind 82% der Umfrageteilnehmer. Als Voraussetzung dafür wird insbesondere eine gute Erkennungsrate ohne bzw. nur mit automatisierter Nachkorrektur angegeben. Zudem müssten die Arbeiten gut in den bestehenden Digitalisierungsworkflow integrierbar sein und finanziell (von der DFG) gefördert werden. Bevorzugt wird die Prozessierung eigener Bestände bzw. Digitalisate, da bei diesen ein gewisser Qualitätsstandard sichergestellt ist. 46% der befragten Bibliotheken würde ggf. zusätzlich fremde Bestände prozessieren.

An eine OCR-Software werden dabei hohe Anforderungen gestellt. Insbesondere sollte diese neben der bereits erwähnten guten Erkennungsrate in der Bedienung (auch durch die Anbindung an bestehende Workflows) einfach und sowohl kosten- als auch zeiteffizient sein. Wünschenswert ist darüber hinaus eine integrierte Layout- und Strukturerkennung sowie die Möglichkeit zum weiteren

Training der zur Verfügung gestellten Modelle. Einzelne Bibliotheken sehen zudem eine aktive Nutzer-Community sowie die gesicherte Weiterentwicklung der Software als wichtig an. Diese Wünsche decken sich größtenteils mit den Anforderungen der Pilotbibliotheken, die die OCR-D-Software bereits getestet haben.[55]

6 Ausblick

Im Frühjahr 2020 wurde der Prototyp der OCR-D-Software fertiggestellt, sodass eine funktionsfähige Software-Suite für die Volltexterkennung historischer Drucke vorliegt. Die Anforderungen der Bibliotheken an eine OCR-Software werden von OCR-D bereits jetzt gut abgedeckt bzw. sind in den Projektplänen vorgesehen und zeigen das Potential dieses Frameworks. Insgesamt sollte die Software noch – z.B. mit Blick auf deren Durchsatz und Robustheit – optimiert und in weiteren bestandshaltenden Einrichtungen getestet werden, bevor die Volltexttransformation der VD produktiv durchgeführt werden kann.

Literaturverzeichnis

Baierer, Konstantin; Dong, Rui; Neudecker, Clemens (2019): Okralact – a multi-engine Open Source OCR training system, In: *HIP '19. Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*, 25–30. Verfügbar unter doi:10.1145/3352631.3352638.

Englmeier, Tobias; Fink, Florian; Schulz, Klaus U. (2019): „AI-PoCoTo: Combining Automated and Interactive OCR Postcorrection." In: *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 19–24.

Gesellschaft für wissenschaftlich Datenverarbeitung mbH Göttingen (2020): GWDG CDSTAR. Verfügbar unter https://cdstar.gwdg.de/.

GitHub, Inc. subugoe/OLA-HD-IMPL (2020): Implementation of OLA-HD (Ein OCR-D Langzeitarchiv für historische Drucke). Verfügbar unter https://github.com/subugoe/OLA-HD-IMPL.

Github. Workflow-configuration (2020): Verfügbar unter https://github.com/bertsky/workflow-configuration.

Huang, Gao; Liu, Zhuang; van der Maaten, Laurens; Weinberger, Kilian Q. (2019): Densely Connected Convolutional Networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2261–69.

Neudecker, Clemens, Konstantin Baierer, Maria Federbusch, Matthias Boenig, Kay-Michael Würzner, Volker Hartmann, and Elisa Herrmann (2019): OCR-D: An end-to-end open source OCR framework for historical printed documents. In: *Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, 53–58.

Niedersächsische Staats- und Universitätsbibliothek Göttingen (2020): Kitodo – Digital Library Modules. Verfügbar unter https://www.sub.uni-goettingen.de/digitale-bibliothek/digitale-werkzeuge/goobi-digital-library-modules/.

[55] Hier ist anzumerken, dass sechs der neun Pilotbibliotheken ebenfalls an der VD-Umfrage teilgenommen haben.

PID Consortium (2020): ePIC Persistent Identifiers for eResearch. Verfügbar unter https://www.pidconsortium.net/.

Reul, Christian; Springmann, Uwe; Puppe, Frank (2017): Larex: A semi-automatic open-source tool for layout analysis and region extraction on early printed books. In: Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage, 137–42.

Sachunsky, Robert; Schiffer, Lena K.; Efer, Thomas; Heyer, Gerhard (2018): Towards Context-Aware Language Models for Historical OCR Post-Correction. In: *European Association for Digital Humanities 2018 Conference Abstracts.* National University of Ireland, Galway. Verfügbar unter https://eadh2018.exordo.com/files/papers/92/final_draft/EADH_2018_Proposal_Brief_Final.pdf und https://git.informatik.uni-leipzig.de/ocr-d/poster-eadh2018/blob/master/main.pdf.

Seuret, Mathias; Limbach, Saskia; Weichselbaumer, Nikolaus; Maier, Andreas; Christlein, Vincent (2019): Dataset of Pages from Early Printed Books with Multiple Font Groups. In: *HIP '19. Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*, 1–6. Verfügbar unter doi:10.1145/3352631.3352640.

Verbundzentrale des GBV (VZG) (2020): VD18-Portal. Verfügbar unter https://gso.gbv.de/DB=1.65/.

Wick, Christoph; Puppe, Frank (2018): Fully convolutional neural networks for page segmentation of historical document images. In: *2018 13th IAPR International Workshop on Document Analysis Systems* (DAS). IEEE, 287–92.

Autoren:

OCR-D – Koordinierte Förderinitiative zur Weiterentwicklung von Verfahren der Optical Character Recognition (OCR)

Kontaktadresse:

Bei Nachfragen wenden Sie sich bitte stellvertretend für die OCR-D-Förderinitiative an die

Projektkoordinatorin Dr. Elisabeth Engl

Herzog-August-Bibliothek Wolfenbüttel

Abteilung Neuere Medien – Digitale Bibliothek

Lessingplatz 1

D-38304 Wolfenbüttel

engl@hab.de

Konstantin Baierer, Matthias Boenig, Dr. Elisabeth Engl, Volker Hartmann, Clemens Neudecker

Reinhard Altenhöner, Dr. Alexander Geyken, Dr. Johannes Mangei und Dr. Rainer Stotzka

Prof. Dr. Prof. h.c. Andreas Dengel, Martin Jenckel

Alexander Gehrke, Prof. Dr. Frank Puppe

Stefan Weil

Robert Sachunsky, Lena K. Schiffer, Dr. Maciej Janicki, Prof. Dr. Gerhard Heyer

Dr. Florian Fink, Prof. Dr. Klaus U. Schulz

Dr. Nikolaus Weichselbaumer, Dr. Saskia Limbach, Mathias Seuret, Rui Dong, Prof. Dr. Manuel Burghardt, Dr. Vincent Christlein

Triet Ho Anh Doan, Zeki Mustafa Doğan, Jörg-Holger Panzer, Kristine Schima-Voigt, Prof. Dr. Philipp Wieder